COMPUTER SCIENCE PUBLICATION

A STRUCTURED KNOWLEDGE REPRESENTATION SCHEME FOR NATURAL LANGUAGE PROCESSING

B. Cheung and K P. Chow

Technical Report TR-89-05

April 1989

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF HONG KONG
POKFULAM ROAD
HONG KONG

A Structured Knowledge Representation Scheme for Natural Language Processing*

B. Cheung and K. P. Chow
Department of Computer Science
University of Hong Kong

ABSTRACT

major tasks in natural language processing include understanding, question answering, inferencing, and summarizing. A good knowledge representation scheme will facilitate performing these tasks. We present here a representation which is a semantic network consisting of three levels of abstraction: attribute list, case frame and coherent relation. The attribute list stores information related to a single concept. The case frame contains information associated to an event or a state. Relationships amongst states/events are stored in the coherent relations. Applications of the three levels structure to question-answering and summarization will be given.

* The research is partially supported by the University of Hong Kong Strategic Research Grant.

1. Introduction

Major tasks in natural language processing include understanding, question answering, inferencing, and summarizing. A good knowledge representation scheme will facilitate performing these tasks. When analyzing a text in natural language, one needs to translate the text into an internal representation. In question-answering phase, depending on the question type, one needs to extract information from the properties of a concept, find the status of an event or a state, or determine the causal relation between events or states. During the summarizing phase, one has to judge what is important, what is less important, what is irrelevant, and what connection of one event with the next is brought out. In all these processing phases, a significant amount of world knowledge (or background knowledge) is needed. In this paper, we concentrate on the structured representation of the knowledge.

Our proposed representation is a semantic network consisting of three levels of abstractions, namely *attribute list*, *case frame* and *inter-relation*. Associating with a concept [10], there is a set of properties. It stores information related to an individual concept. The case frame [7] is used to store the information associated to an event or a state. Relationship between events and states are contained in the inter-relations [2, 3, 1, 6]. To retrieve properties of a single concept, such as the colour of a car, the attribute list is searched. For questions related to an event or a state, such as the agent of a certain action, the case frame is accessed. These two levels of representation have been studied extensively in the past [8]. Question types like *yes-no-question*, *what* and *who* can be handled because they do not require information about relations amongst events/states. On the other hand, it is not sufficient in handling a coherent text with these two levels only. Question types such as *how* question and *why* question cannot be handled because causal relations amongst states/events are needed. They are usually obtained from the hidden meaning of the input. Also, summarization cannot be done unless we know the relations amongst events/states. This is the reason why we need the top level abstraction of representation.

An input text is analyzed by the natural language processing system and transformed into an internal form which consists of three levels. The details of the parser and the language understanding system are discussed in [4, 5]. In the next section, the details of individual level are given. After that, applications of the network to question-answering and summarizing are presented. The three levels knowledge representation scheme, the question-answering system and the summarization system are

parts of the natural language processing system **BP** [5].

2. Three Levels Knowledge Representation Scheme

2.1. Attribute List for Concepts

For a concept, says [BOY:*boy#2*], BOY is the corresponding *concept type* while *boy#2* is its corresponding *referent*. Each concept instance† has a unique referent. The referent *boy#2* tells that the corresponding concept [BOY:*boy#2*] is the second boy that have occurred. Every concept implicitly asserts the existence of something of the corresponding type. For the bottom layer of abstraction, an attribute list is associated with each concept. Most information of the attribute lists is obtained from adjective words, e.g. the colour of [CAR:*car#13*], the size of [CAKE:*cake#2*] or the age of [BOY:*Tom*]. Since different adjectives may have different semantic aspects [9], a very different set of inferences is resulted, such as the examples in Figure 1.

Figure 1. Sentences exemplify some inferences.

Tom is a young president $\not\Rightarrow$ Tom is young
Mary is older than Ann $\not\Rightarrow$ Mary is old
My towel is wetter than yours $\Rightarrow$ My towel is wet
My car is redder than yours $\Rightarrow$ your car is also red

---

Associating to the concept type of an adjective, there are three semantic features, namely *gradability*, *type of scales* and *relativity* [9]. They are stored in an *adjective information list.* The list contains the following information: gradability, type of scales, dimension of scale, marked adjective, unmarked adjective, positive adjective, comparative adjective, superlative adjective and relativity.

2.1.1. Semantic Features

An adjective is **gradable** if it can be substituted for *A* in the following expressions:

---

†An entity recognized by the image of a concrete concept *c* is called a *instance* of *c*.

*A*er (or: more *A*) than
as *A* as
less *A* than
the *A*est (or: most *A*) of
very *A*

Gradability implies the existence of scale in the semantic structure of the adjective - a scale which grades the relevant dimension. For example, the adjectives "old", "young", "full", "empty", are gradable. The corresponding concept types OLD and YOUNG grades the scale of "age", while FULL and EMPTY grades the scale of "fullness".

There are two **types of scale**: *unary* and *binary*. Scales defined by one adjective is referred to as unary, e.g. red. The corresponding concept type RED grades the scale of "redness". For unary-scale adjectives, "NP_i *is A*" signifies the presence of a property while "NP_i *is not A*" signifies the property is being absent.

Scales defined by pairs of adjectives is called binary. In order to define a binary scale, a pair of adjectives must be:

a) gradable
b) incompatible
c) at least semi-reciprocal

Pairs of adjectives defining binary scales will be symbolized as: $A : A'$. Two adjectives are *incompatible* if they satisfy the following entailment formula:

NP_i is $A \Rightarrow NP_i$ is not A'

Two gradable adjectives are *reciprocal* if they satisfy the following pair of entailment formulae:

NP_i is *A*er than $NP_j \Rightarrow NP_j$ is *A*'er than NP_i
NP_j is *A*'er than $NP_i \Rightarrow NP_i$ is *A*er than NP_j

For example,

Mary is older than Jane ⇒ Jane is younger than Mary
Jane is younger than Mary ⇒ Mary is older than Jane

If for two gradable adjectives only one of the entailment formulae is true, they are known as *semi-reciprocal*. This can be exemplified by

My car is more economical than yours ⇏ Your car is more uneconomical than min
Your car is more uneconomical than mine ⇒ My car is more economical than you

"My car is more economical than yours" does not entail "your car is

more uneconomical than mine'' since both cars may be economical, both uneconomical, or one may be economical and the other is uneconomical.

Depending on whether pairs of adjectives are reciprocal or semi-reciprocal, binary scales are further subdivided into two subclasses: *quasi-antonymic* and *antonymic*. Scales defined by pairs which are only semi-reciprocal is known as quasi-antonymic, e.g. economical:uneconomical, intelligent:unintelligent, efficient:inefficient. Scales which defined by reciprocal pairs of adjectives is known as antonymic. Antonymic scales can be either *asymmetric* or *symmetric*. The asymmetric scales are open at one end but bounded at the other. For an adjective pair which defines an asymmetric scale, one adjective signifies the ZERO-end of the scale, i.e. the absence of the corresponding property. This can be exemplified by the pairs: `smootn:rough`, `dry:wet`, `straight:curved`. The asymmetric antonymic pairs have the following properties:

NP_i is $A \Leftrightarrow NP_i$ is not A'
NP_i is $A' \Leftrightarrow NP_i$ is not A

Symmetric scales are the same at both ends, either *bounded* or *open*. The bounded symmetric scale stretches from one to zero, i.e. from complete presence to total absence of a feature. An example is the the adjectives `empty:full`, which defines a scale bounded by 0 and 1. Sentences like *NP_i is empty* and *NP_i is full* are interpreted as extreme cases. Open symmetric scales are open at both ends and no terms exists which signals the beginning or the end of the scale, e.g `old:young`, `short:tall`, `heavy:light`. The open symmetric scale is the most common binary scales. The open scale antonymic pairs have the following properties:

NP_i is not $A \not\Rightarrow NP_i$ is A'
NP_i is not $A' \not\Rightarrow NP_i$ is A

Binary-scale adjectives have the following characteristic: If an adjective from this class is semantically relevant to one element of a set of entities, then the underlying dimension – and the scale for it – is relevant to every element of the set. For example, all people have some height, all physical entities have some weight.

There are two types of **relativity**, known as the *form relativity* and the *scale relativity*. The relativity of comparative-degree and positive-degree forms of adjectives is known as form relativity. There are six

possible types of form relativity (Table 1).

Type	Implications	Example
Fully Absolute	NP_i is *A*er than $NP_j \Rightarrow NP_i$ is *A* NP_i is *A*er than $NP_j \Rightarrow NP_j$ is *A*	Red
Negatively absolute	NP_i is *A*er than $NP_j \Rightarrow NP_i$ is not *A* NP_i is *A*er than $NP_j \Rightarrow NP_j$ is not *A*	Straight
Semi-absolute	NP_i is *A*er than $NP_j \Rightarrow NP_i$ is *A* NP_i is *A*er than $NP_j \Rightarrow NP_j$ is not *A*	Dry
Weakly absolute	NP_i is *A*er than $NP_j \Rightarrow NP_i$ is *A* NP_i is *A*er than $NP_j \not\Rightarrow NP_j$ is *A* NP_i is *A*er than $NP_j \not\Rightarrow NP_j$ is not *A*	Wet
Semi-relative	NP_i is *A*er than $NP_j \not\Rightarrow NP_i$ is *A* NP_i is *A*er than $NP_j \not\Rightarrow NP_i$ is not *A* NP_i is *A*er than $NP_j \Rightarrow NP_j$ is not *A*	Full
Fully relative	NP_i is *A*er than $NP_j \not\Rightarrow NP_i$ is *A* NP_i is *A*er than $NP_j \not\Rightarrow NP_i$ is not *A* NP_i is *A*er than $NP_j \not\Rightarrow NP_j$ is *A* NP_i is *A*er than $NP_j \not\Rightarrow NP_j$ is not *A*	Old

Table 1. Types of form relativity.

As vagueness and context-dependence are features of natural language, the interpretation of adjectives is a matter of "qualitative" rather than a matter of well-defined "true or false". For example, people may disagree with the examples that have be given above. However, they are based on some elicita$l

Relativity of comparative-degree and positive-degree forms of adjectives shown above is known as form relativity. The sentence "Mary is tall" is interpreted as signifying a high value on the scale of height (more than average height) for human females. Suppose Mary is a child.

The sentence also expresses a true proposition even if Mary's height is only 4 feet, provided most other children of the same age are shorter. Hence, "Mary is tall" is a relative statement, and the relativity may be due to various factors, e.g. Mary may be tall for her age, or tall by the standard of the community she lives in. Moreover, "a tall girl" is not necessarily "a tall person" and certainly differs in height from "a tall tree".

The adjective "tall" exemplifies scale-relativity. Interpreting "tall" as "of more than average height" makes it doubly relative. Since "average height" is not an absolute concept, it can only be established for a given reference set. A statement such as "Mary is tall" means "Mary is tall for a *S*" where *S* is the name of the reference set. When such type of sentences are encountered, there is always a pragmatic assumption that the hearer will identify correctly the set *S* of which the referent of the argument is a member.

2.1.2. Transform from Case Frames Structures to Attributes

When reading a piece of text, each sentence will be processed by the parser first. The inference engine will then transform the case frames, which are of concept types corresponding to adjective words, into attributes of attribute lists in order to facilitate making inferences. A set of adjective information lists is associate with a concept type corresponding to an adjective word. Every adjective information list stores the information such as: gradability, scale type defined, types of form relativity, dimension of scale, the marked and unmarked concept types[†] (if present), the corresponding positive, comparative and superlative concept type (if present). For example, the concept type OLDER, which corresponds to the adjective "older" has only one associated adjective information list:

[†]In all open-scale antonymic pairs (and in some other binary-scale pairs) one of the adjectives in the pair is the unmarked term, the other being marked. The adjective "old", for example, is the unmarked term for the pair young:old. The unmarked term has two functions: it can either signal a high value on the scale defined (e.g. "Mary is old"), or it can be value-neutral, in which it represents the dimension as a whole (e.g. "Mary is eighty-five years old.").

OLDER
gradability: true
scale-type: open-symmetric-antonymic
form-relativity: fully-relative
dimension: age
marked: YOUNG
unmarked: OLD
positive: OLD
relative: OLDER
superlative: OLDEST

On the other hands, the concept type **SHORT**, which corresponds to the adjective "short" may have two associated adjective information lists:

SHORT
gradability: true
scale-type: open-symmetric-antonymic
form-relativity: fully-relative
dimension: height
marked: SHORT
unmarked: TALL
positive: SHORT
relative: SHORTER
superlative: SHORTEST

SHORT
gradability: true
scale-type: open-symmetric-antonymic
form-relativity: fully-relative
dimension: length
marked: SHORT
unmarked: LONG
positive: SHORT
relative: SHORTER
superlative: SHORTEST

Background knowledge such as "All people have some height" and "All physical entities have some weight" is stored in the lexicon. This makes the selection of a suitable adjective information list amongst a number of choices possible. Consider the following input frame as an example:

([SHORT]
(AttributeOf: [BOY:*John*]))

The first adjective information list corresponding to the concept type SHORT will be chosen instead of the second one based on the information in the lexicon (i.e. "All people have some height").

2.1.2.1. Positive Form Transformation

When transforming case frame structures corresponding to positive form adjectives into the corresponding attributes, different scale type defined may have different effects on the attribute list. Table 2 gives a brief summary.

Scale Type	Structure	Result
Unary-scale	NP_i is *A* NP_i is not *A*	[> ZERO-dimension] [= ZERO-dimension]
Open-symmetric antonymic	NP_i is *A* NP_i is *A'* NP_i is not *A* NP_i is not *A'*	[$\in$ MANY-dimension] [$\in$ FEW-dimension] [$\notin$ MANY-dimension] [$\notin$ FEW-dimension]
Bounded-symmetric antonymic	NP_i is *A* NP_i is *A'* NP_i is not *A* NP_i is not *A'*	[= ONE-dimension] [= ZERO-dimension] [< ONE-dimension] [> ZERO-dimension]
Asymmetric antonymic	NP_i is *A* NP_i is *A'* NP_i is not *A* NP_i is not *A'*	[> ZERO-dimension] [= ZERO-dimension] [= ZERO-dimension] [> ZERO-dimension]
Quasi-antonymic	Similar to open-scale symmetric	

Table 2. Summary of Positive Form Transformation.

For the semantic interpretation of sentences, the positive degree form of numerical adjectives "many" and "few" are useful concept. They are psychologically simple, and at the same time appropriately fuzzy [9]. On

the scale for a given dimension, MANY-dimension can be represented as an interval, whose value at the lower end is vague, stretches upwards towards the end of the scale (which itself may be ill-defined). The FEW-dimension represents an interval stretching from the beginning of the scale upwards. The upper end is ill-defined, but is a long distance from the end of the scale. The MANY-dimension and FEW-dimension can be extended to cover non-numerical adjectives. The ZERO-dimension shows the absence of the property (or the zero of the scale for a given dimension). The ONE-dimension shows the only maximum point in the scale for a given dimension defined by a pairs of bounded-symmetric antonymic.

We shall illustrate the positive form transformation using the previously example:

([SHORT]
(AttributeOf: [BOY:*John*]))

As SHORT is marked (i.e. the A' in the A:A' pairs) and open-symmetric-antonymic, the attribute list of the concept [BOY:*John*] becomes

{ ... (height (∈ FEW-height)) ... }

2.1.2.2. Comparative Form Transformation

Comparative form transformation consists of two steps. Form relativity tells the positive-degree forms of adjectives that can be implied by a comparative-degree form (refer to Table 1). In the first step, the corresponding attributes are set according to the positive form transformation introduced in the previous section.

Consider the example "This towel is drier than that towel". The following case frame is input to the inference engine:

([DRIER]
(primum-comparationis [TOWEL:*towel#1*])
(secundum-comparationis [TOWEL:*towel#2*]))

Associating with the concept type DRIER, we have an associating adjective information list as follows:

```
DRIER
   gradability: true
   scale-type: asymmetric-antonymic
   form-relativity: negatively-absolute
   dimension: wetness
   marked: DRY
   unmarked: WET
   positive: DRY
   relative: DRIER
   superlative: DRIEST
```

As DRIER is negatively absolute, according to Table 1, we have

NP_i is *A*er than $NP_j \Rightarrow NP_i$ is not *A*

NP_i is *A*er than $NP_j \Rightarrow NP_j$ is not *A*

The attribute lists of [TOWEL:*towel#1*] and [TOWEL:*towel#2*] are modified to reflect that "they both are not dry", i.e. they both are wet. As **DRIER** is the comparative form of the marked **DRY** and asymmetric-antonymic, referring Table 2, the attribute list of the concept [TOWEL:*towel#1*] becomes

{ ... (wetness [> ZERO-wetness]) ... }

and the attribute list of the concept [TOWEL:*towel#2*] becomes

{ ... (wetness [> ZERO-wetness]) ... }

The second step in the comparative form transformation involves setting the relation between the primum-comparationis and the secundum-comparationis. Unmarked/marked signified the high/low value of the scale of dimension. The corresponding attributes are set to signify the higher and lower of the scale of dimension. For the previous example, **DRIER** is the comparative form of the marked **DRY** which signifies the low-value of the scale of dimension `wetness`. The attribute list of the concept [TOWEL:*towel#1*] is:

{ ... (wetness [> $\text{wetness}_{[\text{TOWEL:}towel\#2]}$] [> ZERO-wetness]) ... }

The attribute list of the concept [TOWEL:*towel#2*] becomes

{ ... (wetness [< $\text{wetness}_{[\text{TOWEL:}towel\#1]}$] [> ZERO-wetness]) ... }

2.2. Case Frame Representations

The middle layer of abstraction is the *case frame representation* of the events/states within which the concepts are related together. The

concepts are related through the case slots: Agent, Object, Theme, AttributeOf, StateOf, PreceptOf, Instrument, Departure, Destination, AffectedEntity, Location, Time, Recipient, Beneficiary. From the parser, we obtain a set of case frames. These case frames will then be passed to the inference engine and those case frames corresponding to adjectives will be transformed into attributes of concepts. The remaining are the case frames of the events/states. With the bottom and middle layers, the sentence "Tom moved the red pyramid to the big table" has the following representation:

Figure 2. Examples of the middle and lowest layers.

The lowest layer of abstraction:

Attribute list of the corresponding [PYRAMID:*pyramid#1*] concept:
{ ... (redness [> ZERO-redness]) ... }

Attribute list of the corresponding [TABLE:*table#1*] concept:

{ ... (size [∈ MANY-size]) ... }

The middle layer of abstraction:

([MOVE]
(agent [BOY:*Tom*])
(object [PYRAMID:*pyramid#1*])
(destination [TABLE:*table#1*]))

It is not sufficient in handling a coherent text if we only use the middle and the bottom layers of knowledge representation since we have not yet obtained the hidden meaning of the input and the relations amongst the states/events. Question types like: *yes/no questions*, *what* type questions and *who* type questions can be handled because they do not require information about relations amongst event/state. The more sophisticated question types like: *how* type questions and *why* type questions cannot be handled because they usually need inferential

knowledge, i.e. the relations amongst events/states, in order to obtain the hidden meaning of the input text. Also, we cannot do the summarization unless we have already known the relations amongst the events/states. These are reasons why the top layer abstraction is necessary.

2.3. Concept Coherent Dictionary

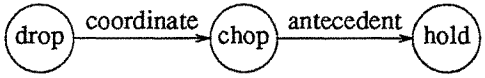
The top layer abstraction is based on the coherence [1, 6]. NEXUS [1] mechanism for generating the relations amongst events/states is simple and efficient. With these relations, information that is explicitly defined in a text can be inferred. On the other hand, the coherent relations used in NEXUS are not ready for doing those question-answering and summarization that requires the causal relations amongst states/events. Questions which require finding the causes and effects, and summarization which requires determining the importancy are unable to be handled by them. A different set of causally relevant coherent relations is used, which makes question-answering and summarizing simple and straightforward, improving the quality of the output, but not introducing computational overload (which may be the case if causal neutral is enforced).

In order to signify the difference, consider the example sentences and the corresponding coherent dictionary in Figure 3.

Figure 3. Examples of causally neutral relations.

1 The peasant was chopping a tree in the woods by the lake. He dropped his axe ...

2 The peasant was chopping wood. When he finished, he dropped his axe.



In the first sentence, the "dropping" disables the "chopping", but for the second sentence, it does not. As the relations are neutral with regards to the causal relationships among "chopping", "holding" and "dropping",

the terms ''chopping'' and ''dropping'' are collected when they appear in a piece of text. The representations of the sentences are identical using Alterman's causal relations. On the other hand, our proposed set is able to distinguish them.

The set of causal relations is based on results from linguistic studies [2,3]. They are also divided into three categories: taxonomic relation (*class/subclass*), partonomic relations (*sequence/subsequence*, *coordinate*), causal relations (*cause*, *enablement*, *reason*, *purpose*).

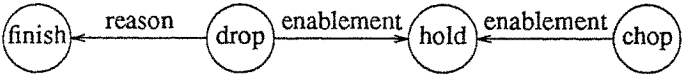
The class/subclass is the property inheritance relation. A class of events inherits both properties and relationships from all its ancestors which are all the nodes that are recursively related to it via the subclass arcs. If one event is a part of another event, and it occurs for a subinterval of time, then the corresponding concepts are in a sequence/subsequence relationship. If an event has parts that occur simultaneously over the same time interval, then the corresponding nodes are in a coordinate relationship. If one event produces another (or makes another event happen), the relationship between their corresponding nodes is classified as cause. If one event makes another event possible but not obligatory, the relationship between their corresponding nodes is classified as enablement. If one event follows as a rational response to another event, the relationship between their corresponding nodes is classified as reason. If one event or situation is planned to become possible via another event or situation, the relationship between their corresponding nodes is classified as purpose.

Figure 4 shows the use of causal relevant relations for the example given in Figure 3.

Figure 4. Causal relevant relations of the example in Figure 3.

Causal relations:

holding an axe **enables** *chopping wood*
holding something **enables** *dropping something*
finished chopping is the **reason** of *dropping the axe*



Consider the question "Why did the peasant stop chopping?". Not "holding an axe" terminates "chopping wood" and "dropping the axe" terminates "holding the axe". This terminated path will be traversed until the end node is arrived. If there is no **reason** for that end node, i.e. "dropping the axe" (as in the first text), we know that it is the ultimate reason that we can obtain (in the first text, the reason is "dropping the axe"). On the other hand, if we have found reason for the end node "dropping the axe" (as in the second text), we may traverse to the corresponding node of the reason to obtain the ultimate reason (so in the second text, the reason is "finished chopping").

2.4. Importancy Factor

In summarization, we have to judge what is important, what is less important and what is irrelevant. In order to facilitate the process of inference, we introduce another type of coherence called the **weight**. A heuristic measure, called *importancy-factor*, is associated with each node within the semantic representation of the text, which signifying the corresponding weight. The values of importancy-factor range from zero to five. Their corresponding meanings range from irrelevant to very important.

The importancy-factors are introduced when constructing the semantic representation. The corresponding value of each node is specified in the constraints associated with the arcs connected to it. Connecting new arcs to an already present node is allowed to alter the corresponding

importancy-factor.

2.5. Implementation of the Three Levels Scheme

Though the three levels correspond to three different conceptual layers of knowledge, they are highly related to each other. For implementation, the set of states/events is represented as nodes of a graph or network. The inter-relations are labeled arcs of the network. For answering questions related to causality, links are traversed. Within each node, a list of case slots is provided to store the information of the state/event. Corresponding to each slot, a concept or another state/event is attached. Each concept is associated with a list of attributes. The information in bottom and middle levels can be retrieved easily.

In order to facilitate locating relevant case frames, the event/state types and also the concept types are organized into a hierarchical structure through the use of supertype and subtype pointers. Moreover, for each event/state type, we have a list of instance pointers pointing to the frames which belong to that type. For each concept types, we have a list of instance pointers pointing to the corresponding concepts. These instance pointers are index for the case frames and concepts which are used in the question-answering system.

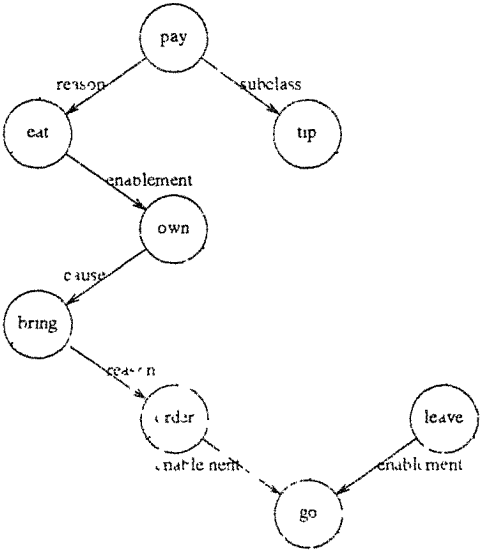
The concept coherent dictionary is a semantic network with nodes being the event/state case frames and arcs being the inter-relationships between the event/state case frames. It is not intended to be a complete explication of the meaning of the event/state terms it contains. Instead, it tries to capture the inter-relations of mutually defined event/state terms. Each term may gain its relative meaning from its position in the network. Nodes are related to one another by one of the seven binary coherent relations.

Attached to each relationship in the dictionary is a set of constraints between the case frames of the terms being related. The constraints are used to control the representation building process. The representation of a text is constructed by matching case frames of the text against the dictionary. A path finding algorithm is used [1]. An arc is being traversed only if it can satisfy the constraints attached to the arc. The representation it produces is a copy of the relevant graph that has been traversed.

Figure 5 gives part of the coherent dictionary. Figure 6 shows a sample text together with its knowledge representation.

Figure 5 Part of the coherent dictionary

Coherent Dictionary:



(Dotted lines represent connections with other parts of the dictionary)

Examples of constraints associated with the relations: (enablement LEAVE GO) -

(match (agent of LEAVE) (agent of GO))
(match (departure of LEAVE) (destination of GO))

(enablement ORDER GO) -

(match (agent of ORDER) (agent of GO))
(match **restaurant** (destination of GO))
(match **food** (object of ORDER))

(reason PAY EAT) -

(match (agent of PAY) (agent of EAT))
(match **restaurant** (location of EAT))

Figure 6 An example

Text:
Mary went to the restaurant and ordered. The waitress brought her a hamburger and she ate quickly. She tipped the waitress and left.
The knowledge representation:

```
{([LEAVE]
    (agent [GIRL·Mary])
    (departure [RESTAURANT:restaurant#1]))
    enablement
    ([GO]
        (agent [GIRL:Mary])
        (destination [RESTAURANT:restaurant#1]))}
{([PAY]
    (agent [GIRL:Mary])
    (dative [WAITRESS:waitress#1]))
    subclass
    ([TIP]
        (agent [GIRL:Mary])
        (dative [WAITRESS·waitress#1]))
    reason
    ([EAT]
        (agent [GIRL:Mary])
        (object [HAMBURGER:hamburger#1])
        (location [RESTAURANT·restaurant#1]))
        enablement
        ([OWN]
            (possby [GIRL·Mary])
            (object [HAMBURGER·hamburger#1])
            (location [RESTAURANT:restaurant#1]))
            cause
            ([BRING]
                (agent [WAITRESS:waitress#1])
                (dative [GIRL:Mary])
                (object [HAMBURGER:hamburger#1])
                (location [RESTAURANT:restaurant#1]))
                reason
                ([ORDER]
                    (agent [GIRL:Mary])
                    (object [HAMBURGER:hamburger#1])
                    (location [RESTAURANT:restaurant#1]))
                    enablement
                    ([GO]
                        (agent [GIRL:Mary])
                        (destination [RESTAURANT:restaurant#1]))}
```

(attribute lists have not been shown)

3. Question-Answering System

The first application we are going to discuss is the question-answering system. The question-answering system is a rule-based system. After a question has passed through the parser, a case frame is produced, e.g., the question "What did Mary order?" is of the following form:

([ORDER
(agent [GIRL:*Mary*])
(object [ENTITY:*])])

The case frame is then matched against the list of case frames through instance pointers. The matched case frames are used as starting points for traversing the network.

According to different question types, we have different methods of traversing the network. Rules are divided into three packets because of their usage and efficiency. The first packet corresponds to extract information from the attribute list level and case frame level. It is used to answer questions of types *who, whom, what, which* and *yes/no*. For question type *why* and *how*, causal relation between states is needed to obtain an answer. The rules in the second and third packet are designed for this purpose.

3.1. Packet One

The Packet One mainly concerns with question types *who*, *whom*, *what*, *which* and *yes-no-question*. These types of questions correspond to locate a concept, retrieve information from the attribute list of a concept, or extract information from a case frame slot. Matching the case frame from the question against the internal representation of a text is sufficient to generate the answer. There are four rules in this packet, two of them are shown below. Figure 7 is a list of sample questions answered by the system.

P1.1 Do the match operation (The list of frames matched successfully will be returned).

If the returned list is not null;

Then they are passed to the text generator as the answer to the question.

P1.2 (The returned list is null)

If there are offspring types for the corresponding question;

Then we can choose the frames from the pointer lists of the offspring types as the candidates and restart this rule packet (i.e. back to rule 1).

P1.1 corresponds to a successful match. The answer can be generated using the information in the matched case frame. For question 1, the case frame representation of the question, as shown in the above section, is matched against the represtation of the text, as shown in Figure 3. A successful match is encountered and an answer is generated. P1.2 is triggered when an unsuccessful match is encountered. Instead of skipping the question, case frames of subtype of the current case frame will be searched, e.g. the question 4 and 5 in Figure 7. In question 5, the action "serve" is not explicitly defined in the text. On the other hand, "bring" is a subtype of "serve", which is searched when rule P1.2 is fired. Question 4 is another example of searching the subtype.

Figure 7. Example Questions for Packet One.

Question 1: What did Mary order?
Answer: Mary ordered a hamburger.

Question 2: Did Mary leave the restaurant?
Answer: Yes, she left the restaurant.

Question 3: What did Mary eat?
Answer: Mary ate a hamburger.

Question 4: What had Mary done?
Answer: Mary went to the restaurant. She ordered a hamburger. She ate the hamburger. She tipped the waitress. She left the restaurant.

Question 5: Who served Mary with the hamburger?
Answer: It is the waitress who served Mary with the hamburger.

3.2. Packet Two

The Packet Two handles question type *why*. For an incoming question of type *why*, it is transformed into a case frame tagged with type *why*. For example, the question "Why did the waitress bring Mary the hamburger" is represented as

([BRING
(agent [WAITRESS: *waitress#i*])
(recipient [GIRL: *Mary*])
(object [HAMBURGER: *hamburger#i*])])

The case frame of the question is then matched against the list of case frames through instance pointers. The matched case is then used as the starting point, follow the subsequence, reason or purpose links, and the target case frames are the answer. Following is some of the rules:

P2.1 Do the match operation (The list of frames matched successfully will be returned).

If the returned list is null and there are offspring types for the corresponding question;

Then we can choose the frames from the instance pointer lists of these offspring types as the candidates and restart this rule packet (i.e. back to the beginning of rule1).

P2.2 **If** the returned list is null and no corresponding offspring types exist;

Then we can signal "I haven't been told".

P2.3 **If** the answer is implicitly implied from the matched frames (e.g. with case relation like *in order to*, or the matched frame is nested in another frame and connected by case relation *by means of*, etc)

Then a selected subgraph will be passed to the text generator as the solution.

P2.4 **If** there are concept coherent relations with type reason pointing out from the matched frames (i.e. the corresponding frames are the reason of the matched frames);

Then the corresponding frames of these relations will be passed to the text generator as the solution.

P2.5 **If** there are concept coherent relations with type purpose pointing out from the matched frames (i.e. the corresponding frames are the purpose of the matched frames);

Then the corresponding frames of these relations will be passed to the text generator as the solution.

Rules P2.1 and P2.2 are used to handle successful match and unsuccessful matches. Rule P2.3 corresponds to case frame which contains a slot storing the reason of the event or state. Rules P2.4 and P2.5 are used to following pointers along the links reason and purpose. In order to answer the above question with respect to the text on Figure 6, rule P2.5 is fired and the target case frame is used to generate the answer:

([ORDER
(agent [GIRL:*Mary*])
(object [HAMBURGER:*hamburger#1*])
(location [RESTAURANT:*restaurant#1*])])

The answer is:

Answer: Because Mary ordered it.

3.3. Packet Three

The third packet concerns with handling question type *how*. The activation of these rules is similar to rules in the Packet Two. The difference is that the link following is performed in the opposite direction since we are determining the effect instead of the cause. Currently, there are around ten rules in this packets and some of them are shown below:

P3.1 Do the match operation (The list of frames matched successfully will be returned).

If the returned list is null and there are offspring types for the corresponding question;

Then we can choose the frames from the instance pointer lists of these offspring types as the candidates and restart this rule packet (i.e. back to the beginning of rule1).

P3.2 **If** the returned list is null and no corresponding offspring types exist;

Then we can signal "I haven't been told".

P3.3 **If** the answer is implicitly implied from the matched frames (e.g. with case relation like *by means of*, or the matched frame is nested in another frame and connected by case relation *in order to*, etc.);

Then a selected subgraph will be passed to the text generator as the solution.

P3.4 **If** there are concept coherent relations with type purpose pointing to the matched frames (i.e. the matched frames are the purpose of the corresponding frames);

Then the corresponding frames of these relations will be passed to the text generator as the solution.

P3.5 **If** there are concept coherent relations with type subsequence pointing out from the matched frames;

Then the corresponding frames of these relations will be passed to the text generator as the solution.

Since their structure is very similar to rules in Packet Two, we shall not give any further explanation.

4. Automatic Summarizing System

The Summarizing System is another rule-based system. The input to the system is the three levels structure of the original text while the output is an ordered list of case frames which will then passed to the text generator. It is based on the following three hypothesis:

1 The input to the Summarizing System composed of structured chunks of concept coherent event/state frames (a connected graph).

2 In order to produce high quality output in terms of semantics, we must try to traverse the whole semantic representation of the text and make a selection amongst all the nodes based on rules of the system. The rules are constructed through a careful study of the concept coherent relations.

3 Besides doing the selection of frames, the rules also determine the order in which they are to be presented based on the coherent relations.

When doing summarization, the graph will firstly undergo a topological sort into a number of directed trees. They will then pass into the rule-based system. The trees are then traversed starting from the root nodes. Nodes are selected based on the types of the relation together with the importancy factor. For example, the subtrees rooted with nodes into which the **subclass** or **subsequence** arcs enter will not be selected because details are suppressed. This is illustrated by rules S.2 and S.3 below. During processing, the node that is currently visit is called the *current node* and the selected nodes are stored in an *ordered list*. There are around fifteen rules and some of them are as follow:

S.1 **If** the whole concept coherent chunks have been traversed;

Then return the *ordered list* as the output of the system (which will then passed to the text generator).

S.2 **If** the *current node* has some **subclass** arcs coming out from it;

Then the corresponding nodes incident by these **subclass** arcs will not be chosen (During summarizing, we need not give the details).

S.3 **If** the *current node* has some **subsequence** arcs coming out from it;

Then the corresponding nodes incident by these **subsequence** arcs are no need to be visited.

S.4 **If** the *current node* has some **reason** arcs coming out from it and there is a chain of **reason** arcs which finally result in a node into which a **subclass** or **subsequence** arc enters;

Then the corresponding subtrees will not be visited (As the whole chunk of subtree is just designated to explain the corresponding node from which the corresponding **subclass** or **subsequence** arc leaves).

S.5 **If** the *current node* has a chain of **reason** arcs coming out from it and it had not marked *unprinted;*

Then some of the nodes following the long chain will be selected into the *ordered list* (based on the importancy factor), also the *current node* will be entered into the *ordered list* and a new *current node* will be chosen (Node that the reason of

something is given first).

Since the rules are in English form, we shall not give additional details here. For detail discussion, please refer to Cheung's thesis [5]. Figure 8 contains some sample runs of the Summarizing System.

Figure 8. Examples for the Summarizing System.

1 Mary went to the restaurant and ordered. The waitress brought her a hamburger and she ate quickly. She tipped the waitress and left.
Summary: Mary went to the restaurant and she ate a hamburger.

2 Mary went to the restaurant and ordered. The waitress brought her a hamburger and she was angry because what she had ordered were sandwiches.
Summary: Mary went to the restaurant. She ordered sandwiches but the waitress brought her a hamburger.

---

5. Conclusion

This paper has introduced a technique of implementing a well-structured three levels knowledge representation scheme. By using attribute lists, the lowest level of abstraction handle the attributes of the concepts which are related by a case frame. The middle level of abstraction is the case frames representation of the the events/states. The highest level of abstraction handles the inter-relations amongst events/states. The technique of concept coherent dictionary has been employed. In order to have the ability to handle the different causal aspects of the relations, the design of the dictionary needs a great deal of creativity and is in fact an art.

References

1. Richard Alterman, A Dictionary Based on Concept Coherence, *Artificial Intelligence* **25** (1985), 153-186.
2. Robert de Beaugrande and Wolfgang Dressler, *Introduction to Text Linguistics,* Longman, London (1981).
3. Gillian Brown and George Yule, *Discourse Analysis,* Cambridge University Press, Cambridge (1983).
4. B. Cheung and K.P. Chow, Universal Feature Instantiation Principles and Wait-And-See Strategy, pp 501-506 in *Proceedings of the Internationl Computer Symposium 1988* , Taipei, Taiwan (Dec 1988).
5. B. Cheung, A Concise Framework of Natural Language Processing, MPhil Thesis, Department of Computer Science, University of Hong Kong, Hong Kong (1989).
6. Dan Fass, Collative Semantics: An Approach to Coherence, Memoranda in Computer and Cognitive Science (MCCS-86-23), Computing Research Laboratory, New Mexico State University (1986).
7. C. Fillmore, The Case for Case, in *Universals in Linguistic Theory* , Holt, Rinehart & Winston, New York (1968).
8. B.J. Grosz, K.S. Jones, and B.L. Webber (Eds), *Readings in Natural Language Processing,* Morgan Kaufmann, Los Altos, CA. (1986).
9. Jan Rusiecki, *Adjectives and Comparison in English: A Semantic Study,* Longman, London (1985).
10. J.F. Sowa, *Conceptual Structures: Information Processing in Mind and Machine,* Addison-Wesley, Reading, MA (1984).